04

Constructor personalizado

Transcripción

[00:00] ¿Qué podría ser eso? Quizás alguna validación extra, como un if, if agencia es mayor que cero, o mejor dicho es menor o igual a cero, entonces imprímeme algún mensaje de error o algo así. Y ahora yo ya estoy validando el valor que agencia va a tener aquí en este momento.

[00:40] Pero si agencia fuera igual a cero, él solamente va a imprimir este valor y bueno, de ahí no va a ser absolutamente nada y de todas formas va a estar con el valor cero aquí adentro y nosotros no queremos eso, pero entonces, ¿qué podemos hacer en ese caso? Podemos decirle: "Oye, al momento de crear una cuenta, infórmame tu número de agencia".

[01:09] Puede ser una regla el negocio también. Entonces, al igual que en cualquier método, yo aquí le doy un int agencia por ejemplo y vemos cómo cambió de azul a plomo. ¿Por qué? Porque yo estoy haciendo referencia a esta variable agencia y no a this.agencia. Vean bien esa diferencia, this.agencia hace referencia a este valor de aquí que automáticamente el ID me ayuda a referenciarlo y la variable solamente agencia hace referencia a esta variable de aquí, ojo con eso.

[01:50] Entonces ahora yo le digo: "Okay, para mi cuenta, tú me vas a informar el número de agencia", y si la agencia fuera menor o igual a cero, voy a imprimir el mensaje de error y solamente voy a poner aquí dentro this.agencia, que es mi variable, va a ser igual digamos al valor por defecto de mis agencias que yo le quiero decir uno, por ejemplo.

[02:16] Le voy a decir ahí en este caso: "si el valor que tú me estás dando es negativo, entonces la agencia de esta cuenta va a ser simplemente la agencia matriz, por ejemplo, la agencia 1". Pero vemos que esta red del negocio ya está siendo validada en el momento del nacimiento de mi objeto cuenta.

[02:35] Ahora, volviendo al código, vemos aquí que cuenta ya no está compilando aquí en este nivel. ¿Por qué? Porque en Java, cuando yo no creo este método de aquí, este constructor como se le llama, yo no necesito crearlo manualmente sino que ya va y lo autogenera por mí, pero en el momento en que yo ya creo mi constructor, le comienzo a enviar parámetros y doy comienzo a personalizar.

[03:08] Ya me dice: "okay, entonces ya no puedo crear un constructor por defecto y tienes que crearlo manualmente", que sería simplemente algo así como cuenta, abrir y cerrar paréntesis y nada más. Volvemos al código de aquí y ya está compilando sin problemas.

[03:28] Ahora, dado que nosotros queremos proteger nuestra cuenta, de ser creada con agencias negativas, entonces nosotros no vamos a usar este constructor por defecto porque él si permite usar el valor por defecto que es cero, y lo vamos a dejar solamente con nuestro constructor usando la agencia. ¿Esto qué significa? Nosotros estamos de cualquier forma obligados a decirle a este objeto al momento de su creación informarle un número de agencia.

[04:04] Por ejemplo, este tipo de situaciones en el cual nosotros le informamos un valor a un objeto, en el momento de su creación, puede ser en aquellas librerías que sirven para leer archivos. Cuando tú llamas a un lector de archivos, tú necesitas informarle el nombre del archivo y él toma ese nombre del archivo en su constructor, y a partir de eso, él añade cierta lógica, cierta validación, entonces aquí nosotros le podemos decir: "Dame el número de cuenta -33 por ejemplo".

[04:42] Y ya le informé el número de cuenta, si yo le doy un getAgencia, voy a imprimir, le do que sí, y me va a decir que su agencia es 1. ¿Por qué? Porque yo ya le estoy diciendo aquí que si es menor que cero entonces me lo cree con la agencia 1 que es la agencia por defecto que vamos a usar. Caso contrario, vamos a crearlo con la agencia 555 y, oh, maravilla, está obviamente en cero. ¿Eso por qué creen que puede ser?

[05:18] Hay un detalle que no hemos visto ahí y es exactamente eso. Nosotros no le estamos asignando ningún valor a agencia en este momento porque nosotros usábamos los setters, pero como lo estamos haciendo a nivel de constructor, caso de que agencia sí sea mayor o igual a cero, perdón, mayor a cero, nosotros le vamos a decir entonces this.agencia es igual a ¿qué? A agencia.

[05:53] Espacio, agencia. Perfecto. Ya ven cómo aquí tenemos esta validación, guardamos aquí, volvemos, ejecutamos y listo, tenemos el número de agencia ya configurado aquí en nuestra cuenta. Ahora ustedes se preguntarán. ¿Pero Diego, no sería más fácil usar setters y todo eso? Quizás sí. ¿Pero cuál es la ventaja de constructor?

[06:22] El constructor te obliga a informar parámetros al momento del nacimiento del objeto. El constructor te fuerza a dar información para que este objeto sea creado, por lo tanto yo aquí puedo asegurarme que ninguna cuenta va a ser creada sin un número de agencia válido por ejemplo, lo cual es muy importante. Toda cuenta debe pertenecerle a una agencia. En el caso del setter es opcional, y aquí llega otro punto muy importante.

[06:53] ¿Será que entonces seguimos necesitando el setter? Recuerdan cuando la clase pasada les dije que dependía mucho del negocio qué métodos debíamos implementar? Pues en este caso tenemos que volver a tener en cuenta eso. Si yo ya le estoy informando la agencia aquí, ¿será que yo necesito aquel método set? ¿Será que yo debería poder cambiar la agencia en cualquier momento?

[07:23] Como les dije, eso va a depender mucho de las reglas del negocio. Por ejemplo, si yo le digo: "Bueno, puedo hacer transferencia de una cuenta entre varias agencias", el método set es totalmente válido. Ahora, si yo le dijera: "No, cada cuenta es creada para una agencia", y si esa cuenta quiere ser de otra agencia, ahí se crea otra cuenta nueva. Ahí ya no aplica este método. ¿Por qué?

[07:54] Porque por regla del negocio yo no puedo editar la agencia. ¿Entonces qué tengo que hacer? Simplemente borro, vengo aquí y elimino el método set.agencia. De esta forma nuestra información de agencia, nuestro campo agencia ya está totalmente aislado y solo es manipulable al momento de la creación del objeto en sí. Este es el poder del constructor. Nos fuerza, nos obliga a dar información para que nuestras reglas del negocio funcionen correctamente.